Theoretical Computer Science 23 (1983) 305–315
North-Holland Publishing Company

COMPUTATION OF RECURSIVE FUNCTIONALS USING MINIMAL INITIAL SEGMENTS*

Dan GORDON
Department of Computer Studies, University of Haifa, Haifa 31999, Israel

Eliahu SHAMIR
Institute of Mathematics, The Hebrew University, Jerusalem, Israel

Communicated by A. Meyer
Received May 1979
Revised May 1982

Abstract. The following problem in the computation of partial recursive functionals is considered: Minimizing the length of inital segments of input functions containing all function values requested by a machine computing a partial recursive functional. A recursive functional F is constructed such that any algorithm for F has unbounded redundancy, i.e. it requests function values on inputs unboundedly larger than those on which the output of F depends. However, for any recursive functional F such that the length of the segment on which F depends is itself a recursive functional, a non-redundant machine for F can be effectively constructed. Also considered are machines on 0–1 sequences for which it is shown that a machine realizing a given level of significance in a universal test of randomness must have unbounded redundancy.

1. Introduction

The computational complexity of functionals has been studied by Constable [3], Lynch [9] and Symes [12]. However, in such computations there is another very natural measure to be considered, namely, the 'amount of information' requested by the computing device from its input or oracle, and it is reasonable to try and minimize that information. The type of 'information measure' dealt with in [5, Ch. 1] and here is the length of the initial segments used as input, although other types, mentioned at the end, might also be considered.

This problem is of some practical interest for several reasons:

(1) Input operations in a computer are usually time-consuming.

(2) Raw data for some computations may be expensive or hard to get.

(3) In a multi-processor environment some processors may be waiting for redundant information from other processors. By striving to use minimal information the

* Research based on Chapter 1 of the first author's D.Sc. thesis in mathematics carried out under the second author's supervision at the Technion-Israel Institute of Technology.

0304-3975/83/0000-0000/$03.00

overall computation could be speeded up by reducing the waiting time of some processors and, indirectly, by reducing the load on the communication links.

Let $\mathbb{N}$ denote the set of natural numbers, $\mathcal{F}$ the set of all total functions from $\mathbb{N}$ to $\mathbb{N}$. Elements of $\mathcal{F}$ will be denoted by $f, g, h, \ldots$ and may be regarded as ω-type sequences of integers $g = (g_0, g_1, \ldots, g_n, \ldots)$ where $g_n = g(n)$. A partial functional F is a function with domain $\mathcal{F}$ and range $N \cup \{\omega\}$. A partial functional F is called total (or simply a functional) if for every $g \in \mathcal{F}$, $F(g) \in \mathbb{N}$. When $F(g) = \omega$, we say that $F(g)$ is undefined.

There are various alternative definitions of partial recursive functionals, which, because the domain consists only of total functions, can easily be shown to be equivalent. The reader is referred to [10, p. 358] for a discussion of functionals.

Let $\varphi_0, \varphi_1, \varphi_2, \ldots$ be an acceptable Gödel enumeration of the partial recursive functionals. For any $i \in \mathbb{N}$ and $g \in \mathcal{F}$, if $\varphi_i(g)$ is defined, then φ_i requests only a finite number of answers from the oracle. This leads to:

Definition 1.

$$L_i(g) = \textbf{if } \varphi_i(g) \in \mathbb{N} \textbf{ then } \max\{n \mid \varphi_i \text{ requested } g_n\} + 1 \textbf{ else } \omega.$$

$L_i(g)$ is the length of the shortest initial segment of g which contains all requests of φ_i from g. If $\varphi_i(g)$ is undefined, then $L_i(g)$ is undefined (even though the number of requests may be bounded). L_i is clearly a partial recursive functional.

We consider the following problem: Given a partial recursive functional F, does there exist an index i for F (i.e., $\varphi_i = F$) such that L_i is minimal? (i.e., if j is another index for F, then for any $g \in \mathcal{F}$, $L_i(g) \leq L_j(g)$.) This motivates the following definition:

Definition 2. If F is a partial recursive functional, we define the *dependence functional* F' of F as:

$$F'(g) = \textbf{if } F(g) \in \mathbb{N} \textbf{ then } \min\{n \mid (\forall f \in \mathcal{F})(f|_n = g|_n \to F(f) = F(g))\}$$

$$\textbf{else } \omega.$$

where $g|_n = (g_0, g_1, \ldots, g_{n-1})$.

$F'(g)$ is the length of the shortest initial segment of g on which the value $F(g)$ (if defined) actually depends. It is also an obvious lower bound for $L_i(g)$.

Lemma 1. *If $\varphi_i(g) \in \mathbb{N}$, then $L_i(g) \geq \varphi_i'(g)$.*

Definition 3. φ_i is a *minimal* (or non-redundant) algorithm if, for every $g \in \mathcal{F}$, if $\varphi_i(g) \in \mathbb{N}$, then $L_i(g) = \varphi_i'(g)$.

If $\varphi_i(g) \in \mathbb{N}$, then the minimal algorithm uses the smallest initial portion of the input it possibly can and still be well defined. There are two main results concerning

the existence of minimal algorithms. Firstly, we construct a total recursive functional F such that any algorithm for F uses unboundedly redundant segments from some inputs (i.e., if $\varphi_i = F$, then for any k, $\exists g \in \mathscr{F}$ such that $L_i(g) > F'(g) + k$). Secondly, we show that if both F and F' are partial recursive, then a minimal algorithm for F can be effectively constructed from any given algorithms for F and F'.

The results obtained are machine-independent, but they are obviously more relevant to computing devices whose input (or oracle) is in the form of an ω-type (semi-infinite) tape on which the sequence $g = (g_0, g_1, \ldots)$ is written. Such machines can be assumed to read the numbers in sequence (first g_0, then g_1, etc.). The only technical assumption we make is that a request for input is considered as at least one step in a computation. We use the convention that $F = G$ ($F \leq G$) means that for every $g \in \mathscr{F}$, either $F(g) = G(g) = \omega$, or $F(g)$, $G(g) \in \mathbb{N}$ and $F(g) = G(g)$ ($F(g) \leq G(g)$). The word 'recursive' is usually used for subsets of $\mathbb{N}$, but we shall use it also to describe subsets of $\mathscr{F}$ whose characteristic functional is recursive, as in [10, p. 336]. We denote the length of a finite sequence t by $|t|$.

2. Main results

Theorem 1. *For every total recursive functional H ($H \geq 1$), there exists a total recursive functional F with range $\{0, 1\}$ such that if $\varphi_k = F$, then the set*

$$\mathscr{C}_k = \{g \in \mathscr{F} \mid L_k(g) \geq H(g) \text{ and } F'(g) = 1\}$$

is not recursive.

Proof. Let R be an r.e. non-recursive set and choose a standard enumeration of R. Let $\varphi_i = H$ and $G = H + L_i$. Define

$F(g) =$ **if** g_0 does not appear within $g(G(g))$ steps in the enumeration of R **then** 0 **else** 1.

For every $g \in \mathscr{F}$, we claim

(1) $g_0 \notin R \Leftrightarrow F'(g) = 1$ and

(2) $g_0 \in R \Leftrightarrow F'(g) = G(g) + 1$.

Proof of (1): If $g_0 \notin R$, then $F(g) = 0$ regardless of the values of $g_1, g_2, \ldots$ so $F'(g) = 1$. To see $\Leftarrow$, assume $F'(g) = 1$, so for all $n \geq 1$, $F(g)$ does not depend on g_n. Assume $g_0 \in R$ and that g_0 appears after m steps of enumerating R. Let $n = G(g)$ and g^1, g^2 be the sequences obtained from g by changing g_n to 0 and m respectively. It is simple to verify that $G'(g) < n$ so $G(g^1) = G(g^2) = G(g) = n \Rightarrow F(g^1) = 0$ and $F(g^2) = 1$, contradicting $F'(g) = 1$.

Proof of (2): If $g_0 \in R$ then, by arguments similar to the above, $F(g)$ depends on $g(G(g))$ and not on anything beyond, so $F'(g) = G(g) + 1$. $\Leftarrow$ follows from (1). Note that since $H \geq 1$, we also have $g_0 \in R \Leftrightarrow F'(g) > 1$.

Let $\varphi_k = F$, then the set $\mathscr{A} = \{g \mid L_k(g) \geq H(g)\}$ is recursive, the set $\mathscr{B} = \{g \mid F'(g) > 1\}$ is not recursive and, by Lemma 1, $\mathscr{B} \subseteq \mathscr{A}$. Therefore $\mathscr{C}_k = \mathscr{A} - \mathscr{B}$ is not recursive. □

The following concept will be useful here and later on: $\mathscr{A} \subseteq \mathscr{F}$ is called *initially-finite* if $\{g_0 \mid g \in \mathscr{A}\}$ is finite. Otherwise we call $\mathscr{A}$ *initially-infinite*. Note that $\mathscr{C}_k$ of Theorem 1 is initially-infinite.

Corollary 1 (The Unbounded Redundancy Theorem). *There exists a total recursive functional F with range $\{0, 1\}$ such that if $\varphi_k = F$, then $L_k - F'$ is unbounded. In fact, $L_k - F'$ is unbounded on any initially-infinite subset of $\mathscr{C}_k$.*

The corollary follows by choosing $H(g) = g_0 + 1$ in Theorem 1. This result strengthens [5, Theorem 1.2] which only showed redundancy.

Is there a minimal algorithm for F when F' is also recursive? The answer seems obvious: Given $g \in \mathscr{F}$, compute $n = F'(g)$ and then use any algorithm φ_i for F on $g|_n$. If φ_i requests more elements, any integer may be fed to it because $F(g)$ depends only on $g|_n$. The problem is that the *entire* procedure has to remain within $g|_n$ even before n is known, so we need an algorithm for F' which remains within $g|_{F'(g)}$ while it is computing $F'(g)$. From Theorem 2 below it will follow that such an algorithm for F' can be effectively constructed from any given algorithm for F'.

Theorem 2. *If G is a partial recursive functional such that $G' \leq G$, then there exists an algorithm φ_i for G such that $L_i(g) \leq G(g)$ for every $g \in \mathscr{F}$ on which G is defined. Furthermore, φ_i can be effectively constructed from any given algorithm for G.*

Proof. Let φ_j be any algorithm for G. Let Λ denote the empty sequence. The algorithm φ_i operates as follows on any input $g \in \mathscr{F}$:

1. Set $k \leftarrow 0$ and $r_0 \leftarrow \Lambda$.
2. Set φ_j working dovetail fashion on all finite extensions of r_k (including r_k itself). If at any time φ_j requests more elements from such an extension than available, pass on to the next computation in the dovetail process.
3. If at step 2 φ_j does not halt with an answer for any finite extention of r_k, then our algorithm never terminates.
   Otherwise:
4. Let t_k be the first extension of r_k on which φ_j halts at step 2, and let $\varphi_j(t_k)$ denote the value obtained (i.e., $\varphi_j(t_k) = G(h)$ for every $h \in \mathscr{F}$ extending t_k).
5. Let $n_k = \min\{\varphi_j(t_k), L_j(t_k)\}$ and $s_k = t_k|_{n_k}$. ($L_j(t_k)$ denotes $L_j(h)$, where h is any extension of t_k.)
6. Start checking if s_k is an initial segment of g by reading one element at a time from g (starting with g_0) and comparing it to the respective element of s_k. If $s_k = \Lambda$ (i.e., $n_k = 0$), it is an initial segment of g and no elements are read.

7. If s_k is an initial segment of g, print $\varphi_j(t_k)$ and halt.
   Otherwise:
8. Let l_k be the length of the minimal initial segment of g we have to read in order to determine that s_k is *not* an initial segment of g. Set $r_{k+1} \leftarrow g|_{l_k}$.
9. $k \leftarrow k+1$ and go to step 2.

The purpose of the algorithm is to find an initial segment of g (of length $\leq G(g)$) on which φ_i is defined. The t_k's, produced by the dovetail process, are finite sequences on which φ_j is defined, but comparing them with g may require redundant elements from g. This is avoided by using at every stage the available value of $G(\varphi_j(t_k))$ as a cut-off. The s_k's are the shortened sequences and they are compared to g one element at a time so that if s_k is *not* an initial segment of g, then only a minimal segment has been read from g ($g|_{l_k}$). $g|_{l_k}$ is r_{k+1} and at stage $k+1$, the dovetail process examines all finite extensions of r_{k+1}. Clearly, in going from r_k to r_{k+1} at least one element is added from g, so $|r_0| < |r_1| < \cdots$, i.e., the t_k's have increasingly long initial segments in common with g. If and when some s_k is found to be an initial segment of g, φ_i outputs $\varphi_j(t_k)$ and halts.

The proof proceeds with three assertions, the first of which shows that the s_k's are sufficient:

(1) If φ_i halts on g, then it prints $G(g)$.

(2) If $G(g) \in \mathbb{N}$, then φ_i halts on g.

(3) If $G(g) \in \mathbb{N}$, then $L_i(g) \leq G(g)$.

Proof of (1): We show that if φ_i reaches step 5 at some stage k and $f \in \mathcal{F}$ is any extension of s_k, then $G(f) = \varphi_j(t_k)$. This clearly implies (1). Let $h \in \mathcal{F}$ be any extension of t_k, then $G(h) = \varphi_j(t_k) \Rightarrow G'(h) \in \mathbb{N}$. $G' \leq G \Rightarrow G'(h) \leq \varphi_j(t_k)$. Now, $L_i(h) \geq G'(h)$ and so $G'(h) \leq \min\{\varphi_j(t_k), L_i(t_k)\} \Rightarrow G'(h) \leq n_k$. Since $f|_{n_k} = s_k = h|_{n_k}$, we get $G(f) = G(h) = \varphi_j(t_k)$

Proof of (2): Assume $G(g) \in \mathbb{N}$ and φ_i does not halt on g. Then either

(a) for some k, φ_i remains at steps 2 and never reaches step 4, or

(b) k increases indefinitely and all the s_k's turn out not to be initial segments of g.

(a) is impossible because every r_k is an initial segment of g and $\varphi_j(g)$ is defined, so if step 2 continued indefinitely it would eventually do a sufficient number of steps on a sufficiently large initial segment of g, and reach step 4. (The initial segment of g would be one of the extensions of r_k.)

To see that (b) is impossible, consider the stage k ($k \geq 1$) at which $|r_k| \geq L_j(g)$. Such a k exists if (b) is true because $|r_0| < |r_1| < \cdots$. Since (a) is ruled out, φ_j must halt at step 2 on some extension t_k of r_k. $|r_k| \geq L_j(g) \Rightarrow L_j(t_k) = L_j(g) \Rightarrow s_k$ is an initial segment of r_k and also of g, so φ_i halts at stage k (or sooner).

Proof of (3): Let k be the stage at which φ_i halts. If $k = 0$, then the initial segment read from g is of length $|s_0| = n_0 \leq \varphi_j(t_0) = G(g)$. Assume $k > 0$. Since $l_0 < l_1 < \cdots$, if we could prove $l_{k-1} \leq n_k$, it would follow that φ_i reads exactly n_k elements from g and $n_k \leq \varphi_j(t_k) = G(g)$, so the result follows.

Assume $n_k < l_{k-1} \Rightarrow t_{k-1}$ extends $s_k \Rightarrow$(by proof of (1)) $\varphi_j(t_k) = \varphi_j(t_{k-1})$. Furthermore, $n_k < l_{k-1} \Rightarrow \varphi_j(t_k) < l_{k-1}$ or $L_j(t_k) < l_{k-1} \Rightarrow \varphi_j(t_{k-1}) < l_{k-1}$ or $L_j(t_{k-1}) < l_{k-1}$ (if $L_j(t_k) < l_{k-1}$, then $L_j(t_{k-1}) = L_j(t_k)$ because t_{k-1} and t_k have $l_{k-1} - 1$ elements in common) $\Rightarrow n_{k-1} < l_{k-1}$, which is impossible.

So φ_i has all the required properties and its description is clearly an effective construction. □

Corollary 2 [5, Theorem 1.5]. *If both F and F' are partial recursive, then a minimal algorithm for F can be effectively constructed from any given algorithms for F and F'.*

Proof. We first show that if $F(g) \in \mathbb{N}$ then $F''(g) \leq F'(g)$, where $F'' = (F')'$. Let $n = F'(g)$ and $f \in \mathcal{F}$ such that $f|_n = g|_n$. Denote $A = \{l \mid (\forall h)(h|_l = g|_l \to F(h) = F(g))\}$ and $B = \{l \mid (\forall h)(h|_l = f|_l \to F(h) = F(f))\}$, so $n = \min A$ and $n \in B \Rightarrow n \geq F'(f) = \min B$. From $n \geq F'(f)$ we can also get $F'(f) \in A \Rightarrow F'(f) = n = F'(g)$. Therefore $F''(g) \leq n$.

Theorem 2 can now be applied to F' in place of G. Using the algorithm for F' constructed in the theorem and the technique mentioned before Theorem 2, we get a minimal algorithm for F. Clearly, our description constitutes an effective construction of the minimal algorithm. □

3. Further results

A note on notation: If $i_1, \ldots, i_k \in \mathbb{N}$ and $g \in \mathcal{F}$, we use $(i_1, \ldots, i_k, g)$ as an abbreviation for $(i_1, \ldots, i_k, g_0, g_1, \ldots) \in \mathcal{F}$. We also write $F(i_1, \ldots, i_k, g)$ instead of $F((i_1, \ldots, i_k, g))$. The next result shows that we cannot effectively pass from an algorithm for F to an algorithm for F' (when F' is recursive) or to a minimal algorithm for F (when it exists). This means that in Corollary 2 we cannot eliminate the need for an algorithm for F'.

Theorem 3. (1) *There is no partial recursive function $r(n)$ such that $\forall n \in \mathbb{N}$, if φ'_n is (partial) recursive, then $r(n)$ is defined and $\varphi_{r(n)} = \varphi'_n$.*

(2) *There is no total recursive function $s(n)$ such that $\forall n \in \mathbb{N}$, $\varphi_{s(n)} = \varphi_n$ and if φ'_n is recursive, then $\varphi_{s(n)}$ is a minimal algorithm.*

Proof. (1) Similar to [4, Theorem 5.1]. Define a sequence of functionals

$$F_i(g) = \textbf{if } \varphi_i \text{ halts on } (0, 0, \ldots) \text{ in exactly } g_0 \text{ steps}$$
$$\textbf{then } 0 \textbf{ else } 1.$$

For every i, either φ_i halts on $(0, 0, \ldots)$ or it does not, so either $F'_i = 1$ or $F'_i = 0$. Therefore F'_i is recursive and $F'_i(g) = 1 \Leftrightarrow \varphi_i$ halts on $(0, 0, \ldots)$. It follows that the

existence of $r(n)$ yields a decision method for the non-recursive set $\{i \mid \varphi_i$ halts on $(0, 0, \ldots)\}$.

Part (2) is a consequence of (1). □

One can ask about various predicates involving F' and their place in the non-recursive hierarchy – the analytical hierarchy because as the syntax of Definition 2 shows, one has to quantify over functions. It follows from that definition that $\{(i, n, g) \mid \varphi_i'(g) \simeq n\}$ is in Π_1^1. We restrict ourselves to one lower bound result about a number predicate.

Theorem 4. *For every $A \in \Pi_1^1$ there exists a partial recursive F such that $A \simeq \{i \in \mathbb{N} \mid F'(i, g) = 1\}$ ($g \in \mathscr{F}$ can be arbitrary because it is superfluous).*

Proof. According to [10, Corollary V, p. 378], there exists a recursive relation $R \subset \mathbb{N}^2$ such that $A \simeq \{i \in \mathbb{N} \mid (\forall f)(\exists k) R(\bar{f}(k), i)\}$, where $\bar{f}(k)$ is the code number of $f|_k$ [10, p. 377]. (The important point here is this form of characterizing members of Π_1^1 and not the precise definition of $\bar{f}(k)$.) The result is obtained by taking

$$F(i, f) \simeq \textbf{if } (\exists k) R(\bar{f}(k), i) \textbf{ then } i \textbf{ else } \omega. \quad \square$$

Conversely, if an algorithm for F and $h_0, \ldots, h_{r-1}$ are given, then $\{i \in \mathbb{N} \mid F'(h_0, \ldots, h_{r-1}, i, g) = r + 1\}$ is clearly in Π_1^1.

Since L_i is a type of 'measure', it seems natural to ask whether it is a complexity measure in the usual sense [2], and if not, whether any recursive relationships exist between L_i and a given complexity measure. To do this, we define a complexity measure for functionals axiomatically with the added natural requirement that no more than $L_i(g)$ elements are required from g in order to determine the complexity of $\varphi_i(g)$.

Definition 4. A complexity measure Φ is an enumeration of partial recursive functionals $\Phi_0, \Phi_1, \ldots$ such that

(1) $(\forall g \in \mathscr{F})(\varphi_i(g) \in \mathbb{N} \leftrightarrow \Phi_i(g) \in \mathbb{N})$;

(2) The functional $C_\Phi(i, n, g) = \textbf{if } \Phi_i(g) = n \textbf{ then } 1 \textbf{ else } 0$ is total and recursive;

(3) $(\exists k)(\varphi_k = C_\Phi \wedge (\forall g)(\varphi_i(g) \in \mathbb{N} \rightarrow L_k(i, n, g) \leq L_i(g) + 2))$.

L_i satisfies the first requirement but not the second so it is not a complexity measure. Another result is the following.

Theorem 5 [5, Theorem 1.7]. (a) *There exists a total recursive functional G such that $(\forall i \in \mathbb{N})(\varphi_i(g) \in \mathbb{N} \rightarrow L_i(g) \simeq G(\Phi_i(g), g)$ for all $g \in \mathscr{F}$ except possibly for an initially-finite subset of $\mathscr{F}$).*

(b) *The above is the only uniform bounding relation between L_i and Φ_i or L_i and φ_i.*

This result is intuitively obvious and the proof is omitted because it follows standard methods as in [7].

4. Algorithms on 0–1 sequences

Denote by 2^ω the set of all infinite 0–1 sequences. We assume that the input consists only of elements of 2^ω. These may be regarded as set-oracles with the characteristic function forming the sequence.

Theorem 6 [5, Theorem 1.8]. *If F is a total recursive functional on 2^ω, then*

(1) *F' is bounded by a constant and recursive;*

(2) *A minimal algorithm for F and an algorithm for F' can be effectively constructed from any algorithm for F.*

Proof. 2^ω can be regarded as the set of infinite branches in the infinite binary tree. Consider the set of branches corresponding to the initial segments used by φ_i for computing $F(g)$ for all $g \in 2^\omega$. Since F is a total recursive functional, each such branch has finite length. By König's Infinity Lemma, it follows that this set is finite. A bound for F' is the length of the longest branch in the set. This fact is stated in [1, Proposition III.21] in terms of the modulus of continuity [10, p. 364; 11, p. 23].

Given any $g \in 2^\omega$, all possible computations of φ_i can be carried out in finite memory before any input is read. From the results of these computations we can determine which elements of g really have to be read in order to know $\varphi_i(g)$. This gives us a minimal algorithm for F and an algorithm for F'. □

Theorem 7. *There exists a partial recursive functional G on 2^ω such that G' is not a partial recursive functional.*

Proof. For $g \in 2^\omega$, define $\bar{g} = \min\{n \mid g_n = 1\}$ ($\bar{g} = \omega$ if g is everywhere 0). Now define

$$G(g) = \textbf{if } \bar{g} = \omega \text{ or } \varphi_{\bar{g}}(g_{\bar{g}+1}, g_{\bar{g}+2}, \ldots) = \omega \textbf{ then } \omega \textbf{ else } 0.$$

$G(g)$ is a partial recursive functional and whenever $G(g) \in \mathbb{N}$, $G'(g) = \bar{g} + 1 \Leftrightarrow \varphi_{\bar{g}}$ is total (on 2^ω). If G' were a partial recursive functional, it could be used to give a decision method for $\{i \mid \varphi_i \text{ is total}\}$ as follows: Given $i \in \mathbb{N}$, we can effectively compute an index n_i such that

$$\varphi_{n_i}(g) = \textbf{if } g_0 = 0 \textbf{ then } 0 \textbf{ else } \varphi_i(g_1, g_2, \ldots).$$

Let $g \in 2^\omega$ be the sequence with $g_{n_i} = 1$ and 0 elsewhere, so $G(g) \in \mathbb{N}$. Now compute $G'(g)$ and we have

$$G'(g) = \bar{g} + 1 = n_i + 1 \Leftrightarrow \varphi_{n_i} \text{ is total (on } 2^\omega)$$

$$\Leftrightarrow \varphi_i \text{ is total (on } 2^\omega). \quad \square$$

Note that for a partial recursive functional F on 2^ω for which F' is also a partial recursive functional, Corollary 2 still holds. Naturally, these results extend to any alphabet of 2 or more symbols.

The existence of non-optimal machines also comes up in the more concrete topic of testing randomness of a $g \in 2^\omega$ [8]. We first define the notion of a test using machines. It is rather different but equivalent to [8]:

Definition 5. A test T is an infiite r.e. set of pairs $\{(m, i_m) | m \in \mathbb{N}\}$ such that if we denote $T_m = \varphi_{i_m}$ and $U_m = \{g|_k | L_{i_m}(g) \leq k\}$, then

(a) The range of T_m is $\{0, \omega\}$ (i.e., only 0 can be printed if it halts);

(b) $U_{m+1} \subseteq U_m$;

(c) U_m contains at most 2^{n-m} sequences of length n for $n > m$.

We say that $g \in 2^\omega$ is *rejected by a test T at the level of significance m* if $g = (x, f)$, where $x \in U_m$ and $f \in 2^\omega$. We say that g *passes* T if for some m_0 (and hence for all $m \geq m_0$), g is not rejected at the level $m_0(m)$; i.e., T_m does not halt on g for $m \geq m_0$.

Definition 6. $g \in 2^\omega$ is random if it passes all tests.

This definition is equivalent to Martin-Löf's definition [8]. As in [8], it is quite simple to construct, by diagonalization, a universal test for randomness T such that $g \in 2^\omega$ passes T iff g is random. The notion of a non-redundant test is natural: To reject a finite sequence x (at level m), T_m should not request information beyond x. For example, frequency tests (which reject a sequence when the frequency of a certain pattern exceeds a given quantity) are clearly non-redundant. However, we show that any T_m of a universal test T has unbounded redundancy. We say that φ_i is *k-redundant* (on 2^ω) if for all $g \in 2^\omega$, $L_i(g) - \varphi_i'(g) \leq k$ whenever $\varphi_i(g) \in \mathbb{N}$. Thus, φ_i is 0-redundant iff it is optimal.

Lemma 3. *If φ_i is k-redundant on 2^ω for some $k \geq 0$ and φ_i maps 2^ω onto $\{0, \omega\}$, then there exists a computable $g \in 2^\omega$ such that $\varphi_i(g) = \omega$.*

Proof. Let $g^0 = (0, 0, \ldots)$. If $\varphi_i(g^0) = \omega$, then we are done. Otherwise, let $l_0 = L_i(g^0) - 1$ and consider all sequences obtained from g^0 by all possible changes of $g^0_{l_0}, g^0_{l_0-1}, \ldots, g^0_{l_0-k}$. (Note that $l_0 \geq 0$ because $\varphi_i(g) = \omega$ for some g.) If φ_i diverges on one such sequence, then again we are done. Else, for at least one sequence g^1, φ_i must read beyond $g^1_{l_0}$, because otherwise $\varphi_i'(g^0) \leq l_0 - k$, contradicting the k-redundancy of φ_i. We can also assume that g^1 is the first on which φ_i reads beyond $g^1_{l_0}$ (in some process where φ_i is set to work dovetail fashion on the sequences obtained from g^0). Let $l_1 = L_i(g^1) - 1$ and continue with g^1 in place of g^0, noting that $l_1 > l_0$. We thus get $g^0, g^1, g^2, \ldots$.

If the above process stops at some stage, then we are obviously done. Otherwise, we can define a computable $g \in 2^\omega$, as follows: Given n, g_n is calculated by simulating φ_i as in the above process until we reach l_j such that $l_j - k > n$, and setting $g_n = g^j_n$.

φ_i does not halt on g because every initial segment of g is also an initial segment of some g^l on which φ_i reads beyond it. □

Theorem 8. *If T is a universal test for randomness, then for all m and k, T_m is not k-redundant, i.e., T_m has unbounded redundancy.*

Proof. If T_m were k-redundant, it follows from Lemma 3 that there exists a computable $g \in 2^\omega$ on which T_m (and hence $T_{m'}$ for $m' \geq m$) does not halt. Therefore g passes the universal test T, and so g is random. But a computable sequence g cannot be random because it defines a test T' which it does not pass as follows: $T'_m(h) =$ **if** $h|_m = g|_m$ **then** 0 **else** ω. □

5. Final remarks

Alternative methods of measuring the amount of information are also possible and we briefly mention some examples:

(1) Assuming the numbers on the infinite tape are represented by some finite alphabet, we can consider the length of the initial sequence of symbols as our measure. This brings us back to functionals on infinite sequences of some finite alphabet, with the following difference: In every sequence, at least two symbols appear infinitely many times. Theorems 6 and 7 still hold when the input is restricted to such sequences.

(2) If we consider $g \in \mathcal{F}$ as a set of pairs, i.e., $g = \{(n, g_n) | n \in \mathbb{N}\}$, we can ask which *subsets* of g – as opposed to initial segments – determine the value of a functional F on g. Combinatorial properties of such subsets were examined in [5, Ch. 2] and [6]. The following minimal-information question also suggests itself: Let $R_i(g)$ denote the subset of g requested by φ_i in the computation of $\varphi_i(g)$ [12]. Does every total recursive functional F have an index i such that for all $g \in \mathcal{F}$, no proper subset of $R_i(g)$ determines the value of $F(g)$? Even if g is restricted to 2^ω, the answer is no, as shown by the example $F(g) =$ **if** $(g_0 = 0$ or $g_1 = 0)$ **then** 0 **else** 1.

(3) Assume that the oracle for g is such that the computing device writes n on a request tape and gets g_n on an answer tape. An appropriate measure of information would be the total number of symbols written or read by the device in requests for information. This particular measure was not studied by the authors, but it seems quite likely that some functionals have only redundant algorithms.

(4) Axiomatic complexity theory suggests phrasing 'axioms for information measures' which the examples considered here should satisfy. It could be theoretically interesting to examine which results remain valid for such abstract information measures.

We conclude with the observation that some of the concepts in [4] can be seen from an 'information' point of view. For example, the 'maximally defined sequential approximation' $\hat{f}$ of a function f represents, at any given time in the computation

of f, the maximal sequential output dependent on the input read so far, as stated in [4, p. 58]. Thus the existence of a recursive f with $\hat{f}$ non-recursive is somewhat analogous to our total recursive functional F with F' non-recursive. Another information-oriented result follows from [4, Theorem 2.8]: If f and $\hat{f}$ are both recursive, then an optimal machine for f can be effectively constructed from any machines for f and $\hat{f}$. (By optimal we mean that if y is an initial segment of the input string, then $\hat{f}(y)$ is produced after reading y and before reading any other symbol.) The setting and proof of this result is basically different from our Theorem 2, but these analogies seem to indicate that the concepts and results we have here could be typical of a wide variety of settings.

Acknowledgment

The authors are indebted to Albert Meyer for a tidier proof of Theorem 1 and to the anonymous referees for suggesting numerous improvements in presentation, the question on upper bounds for F' in the analytical hierarchy and the third example of Section 5. Thanks are also due to Raya Rosenfeld for collaboration on the results concerning universal tests for randomness.

References

[1] J.-P. Azra and B. Jaulin, *Récursivité* (Collection 'Programmation', Paris, 1973).
[2] M. Blum, A machine-independent theory of the complexity of recursive functions, *J. ACM* **14** (1967) 322–336.
[3] R.L. Constable, Type two computational complexity, *Proc. 5th Annual ACM Symposium on Theory of Computing*, Austin, TX (1973).
[4] P.C. Fischer, E.L. Robertson and L.V. Saxton, On the sequential nature of functions, *J. Comput. System Sci.* **13** (1976) 51–68.
[5] D. Gordon, Computability, determining sets and dependence degrees of functionals, D.Sc. Thesis (Hebrew), Mathematics Department, Technion–Israel Institute of Technology, Haifa (1976).
[6] D. Gordon, Minimal determining sets of locally finitely-determined functionals, *Discrete Math.* **29** (1980) 175–190.
[7] J. Hartmanis and J.E. Hopcroft, An overview of the theory of computational complexity, *J. ACM* **18** (1971) 444–475.
[8] P. Martin-Löf, The definition of random sequences, *Information and Control* **9** (1966) 602–619.
[9] N.A. Lynch, Relativization of the theory of computational complexity, Project MAC, TR-99 (1972).
[10] H. Rogers, Jr., *Theory of Recursive Functions and Effective Computability* (McGraw-Hill, New York, 1967).
[11] J.R. Shoenfield, *Degrees of Unsolvability* (North-Holland, Amsterdam, 1971).
[12] D.M. Symes, The extension of machine-independent computational complexity to oracle machine computation and the computation of finite functions, Research Report CSRR 2057, University of Waterloo (1971).